TYÖVUOROJEN VARAUSJÄRJESTELMÄN VERKKOTOTEUTUS

CASE: Farmedico

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Mediatekniikan koulutusohjelma

Riihimäen toimipiste, työn hyväksymispäivä

Jani Takala

HAMK
HÄMEEN AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

Koulutusohjelma
Paikkakunta

Työn nimi	Työvuorojen varausjärjestelmän verkkototeutus
Tekijä	Jani Takala
Ohjaava opettaja	Kauko Ojanen
Hyväksytty	_____._____.20_____

Hyväksyjä

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

RIIHIMÄEN TOIMIPISTE
Mediatekniikan Koulutusohjelma

Tekijä Jani Takala **Vuosi** 2010

Työn nimi Työvuorojen varausjärjestelmän verkkototeutus
CASE: Farmedico

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee Internet-tekniikoita, joita käyttämällä on mahdollista toteuttaa työvuorojen varausjärjestelmä verkkoon. Toteutetun sovelluksen on tarkoitus olla dynaaminen ja näyttävä ollen samalla kuitenkin mahdollisimman kevyt.

Työvuorojen varausjärjestelmä toteutetaan lääkealalla toimivalle henkilöstövuokrauspalveluja tarjoavalle yritykselle Farmedico. Farmedicolla on tarve saada asiakkaidensa vapaat työvuorot työntekijöidensä nähtäville ja varattaviksi Internetiin.

Käytettävät tekniikat ovat suurimmalle osalle Internetin käyttäjistä tuttuja, lukuun ottamatta uudehkoja tekniikoita, joihin lukeutuvat JQuery ja Ajax, jota käsitelläänkin teoriaosuudessa hieman laajemmin. Käytetyt tekniikat ovat valittu niiden tuomien ominaisuuksien mukaan. Toteutus voidaan tehdä muillakin tekniikoilla, mutta esitellyt tekniikat ovat tämän työn toteutuksen kannalta järkevimmät. Tekniikoiden läpikäynti etenee johdonmukaisesti tutuista perustekniikoista (HTML, CSS) uudempiin ja samalla vieraampiin tekniikoihin.

Opinnäytetyössä sovelletaan omaa aiempaa kokemusta uuteen lähinnä kirjojen ja Internetin avulla saatavaan tietoon. Tätä opinnäytetyötä voidaan hyödyntää myös tulevaisuudessa, sillä käytetyt tekniikan tulevat olemaan käytössä vielä pitkän aikaa. Opinnäytetyön lukija saa läpileikkauksen dynaamisten Internet-sovellusten mahdollistaviin tekniikoihin.

Avainsanat Internet, PHP, Ajax

Sivut 27 s, + liitteet 1 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

Riihimäki
Degree programme in Media Technology

Author Jani Takala **Year** 2010

Subject of Bachelor's thesis Internet implementation of a shift reservation system

ABSTRACT

Techniques to develop a shift reservation system onto the Internet are handled in this thesis. The purpose of the developed system is to be dynamic and visually impressive while being as light as possible.

The shift reservation system is going to be developed for a company called Farmedico which operates as a recruitment agency in the pharmaceutical sector. Farmedico has the need to have their clients' free shifts to be seen and reserved on the Internet.

People using the Internet are familiar with most of the techniques covered in the thesis. The more unfamiliar techniques are JQuery and Ajax which are covered in more depth in the theoretical section of the thesis. The techniques are chosen by the properties they offer. The reservation system can be accomplished with many other techniques but the ones covered are the most practical for this purpose. The review of the techniques begins with familiar ones like HTML and CSS and moves logically to more complex and more unfamiliar techniques.

The author's own experience and knowledge is integrated with information from books and the Internet for this thesis. Because the reviewed techniques are going to be in wide use in the future, the thesis contains valid information for the years to come. The reader will obtain basic knowledge about the techniques used in the development of dynamic Internet applications.

Keywords Internet. PHP, Ajax

Pages 27 p + appendices 1 p.

SISÄLLYS

TERMISTÖ

Ajax
Asynchronous JavaScript and XML. Tekniikka, jonka avulla client-puolella mahdollistetaan tietojen hakeminen palvelimella sijaitsevasta tietokannasta. Ajax mahdollistaa Internet-selaimen tietojen päivittämisen ilman sen uudelleen latausta.

Asiakas
Katso Client.

Asynkroninen tiedonsiirto
Ajax:in yleisesti käyttämä tiedonsiirtomenetelmä, joka mahdollistaa käyttäjälle huomaamattoman tiedonsiirron.

Client
Clientillä tarkoitetaan tässä tapauksessa Internet-selainta, joka kykenee verkon yli ottamaan yhteyden palvelimeen ja vastaanottamaan verkkosivuja. *Vertaa Palvelin.*

CSS
Cascading Style Sheets, tyylimäärittely, jonka avulla muokataan Internet-sivun ulkoasua.

DHTML
Dynamic HyperText Markup Language. Kuten HTML, mutta lisänä dynaamisuutta, jolla tarkoitetaan, että Internet-sivulla vaihtuu sisältö ilman sivun uudelleen latausta.

DOM
Document Object Model. Rajapinta, jonka kautta JavaScriptin avulla voi muokata HTML-kodumenttia.

Dynaaminen web-ohjelmointi
Katso DHTML.

HTML
HyperText Markup Language. Internet-sivujen standardoitu merkintäkieli.

JavaScript
Oliopohjainen ohjelmointikieli, jonka käyttö tapahtuu pääosin client-puolella.

JQuery
JavaScript-kirjasto.

MySQL
Suosittu ja peruskäyttäjälle ilmainen kieli relaatiotietokantojen käsittelyyn.

Olio-ohjelmointi
Olio-ohjelmoinnilla tarkoitetaan tapaa rakentaa ohjelma-koodi osiin, jolloin siitä tulee helposti hallittavampi ja muokattavampi. Jaettuja osia

kutsutaan luokiksi ja niiden sisällä olevia olion tietoa käsitteleviä funktioita metodeiksi.

Palvelin
Palvelimeksi kutsutaan tietokonetta, jonka tarkoitus on tarjota sitä käyttäville asiakkaille erilaisia palvelinohjelmistoja. Palvelintilaa voi ostaa maksullisena tai sen voi myös tehdä itse. Maksulliset palvelimet ovat yleensä varmistettuja ja ne ovatkin jatkuvasti päällä.

PDO
PHP Data Object. Tietoturvallisempi vaihtoehto tietokantojen hallintaan.

PHP
Hypertext PreProcessor. Suosittu palvelimelta suoritettava web-ohjelmointikieli.

Server
Katso Palvelin.

SQL
Structured Query Language. Standardoitu kyselykieli relaatiotietokantojen hallintaan.

Synkroninen tiedonsiirto
Tiedonsiirtomenetelmä, jossa käyttäjän täytyy odottaa, että kaikki tieto on siirtynyt palvelimen ja asiakkaan välillä.

XHTML
Kuten HTML, mutta rakenne on tarkempi. Jokainen merkintä täytyy avata ja sulkea oikealla tavalla. XHTML:n tarkoitus onkin, että jokaisella selaimella sivut näyttäisivät samalta. *Vertaa XML.*

XML
Vapaasti merkittävä merkitsemiskieli, jonka rakenne täytyy kuitenkin olla oikeanlainen.

XMLHttpRequest
Ajax:in käyttämä tekniikka, joka mahdollistaa JavaScriptin tehdä asynkronisia pyyntöjä palvelimelle.

1 JOHDANTO

Kehittyvä tietoyhteiskunta ja siihen tottuvat ihmiset tottuvat koko ajan entistä helppokäyttöisempiin ja interaktiivisempiin Internet-toteutuksiin. Ihmisten kiire kasvaa koko ajan ja ajankäytön minimointi ja tehokkuuden maksimointi otetaan huomioon joka seikassa. Mitä nopeampi, sitä parempi kuvastaa hyvin ihmisten ajattelutapaa nykyaikana. Vaatimustason kasvaessa myös tekniikoiden tulee kehittyä ja näin on tapahtunut myös Internet-puolella, jolle kehitetään koko ajan uusia tekniikoita tehostamaan ja nopeuttamaan web-ohjelmointia, kuten myös itse tietoliikenneverkkoa. Hyviä esimerkkejä uusista läpilyöneistä tekniikoista Internet-sivujen toteutuksen puolella ovat JavaScript-kirjasto JQuery sekä Ajax. Kumpikaan tekniikoista ei ole vielä ehtinyt edes viiden vuoden ikään, mutta niiden käyttö alkaa silti olla jo vakiintunutta ja asiakkaat osaavatkin jo vaatia niitä hakiessaan verkkosivuilleen tekijää.

Tämä opinnäytetyö käykin läpi yleisimmät dynaamisten ja interaktiivisten Internet-sivustojen luomiseen käytettävät tekniikat ja tarjoaa niistä kevyen läpileikkauksen. Täydellinen selvitys jokaiseen tekniikkaan ei ilman jättimäisen opuksen kirjoittamista ole mahdollista, eikä tällaisen tekemistä ole kukaan vielä erehtynyt edes yrittämään.

Läpikäynti alkaa kevyesti perus HTML-tekniikasta ja päätyy uudempiin tekniikoihin, kuten JQuery ja Ajax, jota käsitellään hieman laajemmassa mittakaavassa.

1.1 Teorian hyödyntäminen CASE-toteutuksessa.

Esille tulevat tekniikat ovat käytössä sovelluksessa, joka toteutetaan henkilöstövuokrauspalveluja tarjoavalle yritykselle Farmedico. Farmedicolla on tarve saada henkilöstön välitys tapahtumaan sähköisesti Internetissä. Opinnäytetyö ei kerro, miten sovellus toteutettiin vaan miten se tullaan toteuttamaa ja mitä tekniikoita käyttäen.

2 DYNAAMISTEN INTERNET-SIVUJEN TOTEUTUS

Vaikkakin Internetin juuret alkavat jo 1960-luvulta ei sen nykymuoto ole ollut olemassa kuitenkaan kuin 1990-luvun alusta, tarkemmin vuodesta 1991, jolloin HTTP (HyperText Transfer Protocol) keksittiin. Sen tarkoitus on alusta asti pysynyt samana. Pakettien siirtäminen clientin ja web-palvelimen välillä on edelleen sen päätehtävä. (Darie, Brinzarea, Chereches-Tosa & Bucica 2006, 10; Internet - Wikipedia 2010.)

2.1 HTML

HTML (HyperText Markup Language) on Internetin standardoitu merkintäkieli. HTML on kieli, jonka Internet-selain ymmärtää ja jonka avulla teksti ja kuva esitetään web-sivulla. HTML:n avulla ei ole mahdollista tehdä monimutkaisia, dynaamisia tai interaktiivisia sivuja, sillä sen ominaisuudet ovat rajoittuneet vain staattisen sisällön esittämiseen palvelimelta. Kun käyttäjä haluaa esittää uutta sisältöä selaimessa, tulee sivu ladata aina uudelleen HTTP:n avulla. Mikäli sivuille halutaan monipuolisempaa sisältöä, tulee käyttää hyväksi muita tekniikoita, kuten PHP-ja JavaScript-ohjelmointikieliä. (Darie ym. 2006, 10)

2.2 DHTML

Internet-sivut ovat suurimmissa osin staattisia. Staattisella tarkoitetaan, ettei sivulla sen lataamisen jälkeen muutu sisältö lainkaan. Vaihtoehtona ovat dynaamiset sivut, jotka ovat yleistyneet sen jälkeen kun käsite Web 2.0 tuli esille vuonna 2004. Käsitteenä Web 2.0 ei tarkoita juuri mitään, kuten ei myöskään DHTML, eli dynaaminen HTML, jotka molemmat ovat olleet enemmänkin markkinointikäsitteitä. Kärjistettynä DHTML kuitenkin tarkoittaa web-sivun esittämistä niin, että siinä sivun lataamisen jälkeen sisältö muuttuu ilman sen uudelleen lataamista. Tällainen esitystapa on mahdollista käyttämällä JavaScriptiä ja sisältöä, kuten CSS-tyylitiedostoja yhdessä selaimen DOM-objektin kanssa. Jotta tämä esitettävä sisältö saadaan noudettua palvelimelta, tulee hyödyntää tekniikkaa nimeltään Ajax. (Souders 2007, 96,97)

2.3 CSS

CSS (Cascading Style Sheets) on nimensä mukaisesti kaskadinen tyyliohje, joka on kehitetty HTML-dokumentin ulkoasun muokkausta varten. CSS:n avulla on mahdollista määrittää muun muassa dokumentissa esiintyvä fontti, taustan väri ja taulukon reunusten paksuus. CSS:n avulla voi pienellä vaivalla tehdä koko sivua tai vain sen osaa koskevia muutoksia. Teknisesti katsoen CSS ei ole pakollinen käytettävä luotaessa dynaamisia web-sivustoja, mutta se ehdottomasti kannattaa sen tuomien ominaisuuksien ja mahdollisuuksien takia. (Darie ym. 2006, 39; CSS - W3C 2010)

2.4 PHP

PHP eli Hypertext Preprocessor on palvelinpuolen ohjelmointikieli. PHP on web-kehittäjien suosiossa sen helpon opiskelukynnyksen vuoksi. Esimerkkejä ja oppaita on saatavissa lukematon määrä sekä Internetistä kuin myös painettuna tekstinäkin. PHP onkin vuoden 2010 Huhtikuussa neljänneksi suosituin ohjelmointikieli.
(Internet – Tiobe Software, viitattu 8.4.2010)

PHP:n suorittaminen/ajaminen eli näkyminen käyttäjän Internet-selaimessa vaatii, että www-palvelimelle on asennettu PHP-kääntäjä. PHP-kääntäjä onkin asennettuna lähes jokaisen maksullista sivutilaa tarjoavan eli hostin palvelimella. Vuoden 2010 keväällä PHP:n varassa toimivia verkkotunnuksia olikin käytössä kaikkiaan lähes 30 % kaikista maailman verkkotunnuksista.
(Internet – PHP Usage statistics, viitattu 8.4.2010)

PHP:n avulla voidaan tehdä paljon muutakin kuin vain palauttaa HTML-sivuja palvelimelta. Sen avulla kyetään muun muassa suorittamaan monimutkaisia laskentatoimia, manipuloimaan tietokantoja ja hyödyntämään olio-pohjaisen ohjelmoinnin etuja.

PHP ei ole ainoa palvelinpuolen ohjelmointikieli, mutta se on kaikista suosituin. Usein PHP on käytössä yhdessä jonkin tietoja säilyttävän varaston kanssa, kuten MySQL-tietokantojen. Nämä tekniikat yhdessä mahdollistavat suurien tietomäärien käsittelyn ja niiden esittämisen Internet-selaimessa. Esitystapa ei kuitenkaan ole kovinkaan muokattavissa. Sivu on staattinen ja uuden tiedon esittäminen palvelimelta vaatii selaimen uudelleen päivittämisen. (Darie ym. 2006, 10,11)

2.4.1 PHP ja tietokantayhteydet

Tietokannat ovat usein suuressa roolissa, kun tehdään laajempaa verkkosivustoa. Tietokantoihin voidaan tallentaa koko sivun sisältö käyttäjätunnuksista ja salasanoista aina sivulla esitettävään tekstiin tai kuvien tiedostopolkuihin. Ilman kunnossa olevaa tietoturvaa tietokantoihin on kuitenkin todella helppo hyökätä ja siellä oleva arkaluontoinen tieto voi joutua vääriin käsiin.

2.4.2 PHP ja MySQL

SQL tulee sanoista Structured Query Language. Se on kieli relaatiotietokantojen käsittelyyn ja hallintaan. MySQL puolestaan on hallintajärjestelmä SQL-tietokannoille. MySQL-tietokantoja on suosittua käyttää PHP:n kanssa, sillä ne ovat molemmat yksityiskäyttäjälle ilmaisia. (Kolehmainen 2006, 281)

Tietokannat muodostuvat yhdestä tai useammasta taulusta eli relaatiosta. Kaikki tietokantoihin tuleva tieto tallennetaan tauluihin. Taulut muodostuvat yhdestä tai useammasta kentästä. Kenttää luotaessa tulee

määrittää minkätyyppistä tietoa siihen tallennetaan (teksti, luku jne.). Käyttäjä voi muokata tietokantoja joko erillisellä ohjelmalla tai kirjoittamalla SQL-komentoja sopivaan kohtaan koodia. (Kolehmainen 2006, 281)

2.4.3 PHP ja tietoturva

PHP ilman tietoturvallisesti oikeaoppista ohjelmointia on todella suojaton hyökkäyksiä vastaan. Hyökkääjän mahdollisuuksia saada arkaluontoista materiaalia itselleen tai vain huvikseen muokata verkkosivua on lukuisia. Ilman kunnollista tietoturvaa hyökkääjä voi kirjoittaa omaa koodiaan sivuille tai kaapata esimerkiksi session, jonka avulla salasanalla suojatulle sivustolle pääsee sisään.

Suurin osa sivuille tapahtuvista väärinkäytöksistä johtuu huonosti toteutetusta syötteiden tarkistuksesta. Jos käyttäjän antamia syötteitä ei tarkisteta voi mahdollinen hyökkääjä syöttää sivulle mitä tietoja tahansa ja tätä kautta saada arkaluontoista tietoa muun muassa tietokannasta. Esimerkiksi, jos käyttäjä haluaa, että lomakkeen jonkin kentän arvon on tarkoitus palauttaa vain lukuarvoja, tulee siihen syötetty arvo tarkistaa siltä varalta, että se pitäisi sisällään jotain muuta. Käyttäjän tunnistetiedot sisällään pitävät sessiot saadaan puolestaan suojattua esimerkiksi generoimalla session avain uudelleen tietyn aikajakson välein, jolloin vanhetunutta avainta käyttävä ei pääse sisään. (PHP-tietoturvaopas 2010)

PHP:n ja tietokantayhteyksien turvallisuutta voidaan parantaa käyttämällä PHP:n 5.1-versiosta lähtien tuettuna ollutta PDO-luokkaa. PDO (PHP Data Objects) on tietokantojen hallintaluokka, jolla on tuki kaikille yleisimmille tietokannoille. PDO:ssa esimerkiksi injektio-hyökkäyksiltä suojautuminen on toteutettu perinteistä `mysql_` -yhteyttä tehokkaammin. (PHP:PDO - Manual 2010)

2.5 Javascript ja DOM

JavaScript on oliopohjainen ohjelmointikieli, jonka käyttö tavallisesti tapahtuu client-puolella. Se siis upotetaan HTML:n sekaan ja ajetaan suoraan selaimessa, toisin kuin PHP, joka suoritetaan palvelimella. (Darie ym. 2006, 12, 15)

JavaScriptin vahvuus on sen mahdollisuus muokata HTML-dokumenttia. Dokumentin muokkaukseen JavaScript hyödyntää alusta- ja kieliriippumatonta DOM-ohjelmointirajapintaa. (Darie ym. 2006, 15)

DOM-mallin avulla voidaan manipuloida (luoda, muokata, etsiä jne.) XML-tyylisiä dokumentteja, joihin lukeutuu myös HTML. Koska DOM-malli ei ole kieliriippuvainen sitä voidaan käsitellä JavaScriptin, PHP:n C#:n C++:n jne. kautta. (W3C Document Object Model 2010)

Client-puolella DOM ja JavaScript voivat yhdessä muun muassa:

- Manipuloida HTML-dokumenttia samalla kun sitä käytetään.

- Lukea ja jäsennellä palvelimelta saatua XML-dokumenttia.
- Luoda uusia XML-dokumentteja.

Palvelinpuolella DOM ja PHP voivat yhdessä muun muassa:
- Koostaa XML-dokumentteja, yleensä lähettääkseen ne clientille.
- Lukea XML-dokumentteja useasta eri lähteestä (RSS-syötteet).

.(Darie, ym. 2006, 15, 17)

DOM-malli mahdollistaa HTML-sivujen muokkauksen JavaScriptillä. Jokainen sivun osa on myös osa DOM-mallia, jolloin myös sen ominaisuudet ja metodit ovat JavaScriptin käytettävissä. DOM-mallin käsittely on yksinkertaista ja standardoitua.(Darie, ym. 2006, 65,66; Asleson & Scutta 2007, 38-45; W3C Document Object Model 2010)

Jokaista DOM-elementtiä sivulla kutsutaan objektiksi. Taulukko on objekti kuten on myös h1-tagi tai sivun titlekin. Kokonainen sivukin on objekti, jota myös kutsutaan nimellä document. Objektilla voi myös olla jokin arvo eli property. Esimerkiksi sivun titlellä voi olla arvo, joka olisi esimerkiksi ”objekti-mallin esittely”. Tähän voidaan viitata dokumentin kautta parametrilla `document.title`. Taulukon riviin voidaan puolestaan viitata arvolla `table.row`. Näihin arvoihin viittaamalla päästäänkin käsittelemään web-sivua JavaScriptin ja DOM:in kautta. (Videolähde, 2008, AJAX essential training, dom)

Seuraava yksinkertainen esimerkki näyttää kuinka DOM-objektia voidaan käyttää JavaScriptin kanssa:

Luodaan index.html, johon tehdään tyhjä div-elementti, id-arvolla divElement.

```
 1: <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
 2: <html>
 3:   <head>
 4:     <meta http-equiv="content-type"
 5:     content="text/html;
 6:     charset=windows-utf8">
 7:     <title>DOM-esimerkki</title>
 8:
 9:     <script src="jsFunctions.js" type="text/javascript"></script>
10:   </head>
11:
12:   <body onLoad="process()">
13:     List of colors: <br />
14:     <div id="divElement">
15:     </div>
16:   </body>
17: </html>
```

Kuva 1 index.html, html-tiedosto

Html-tiedostoon luotuun div-elementtiin viitataan inner.js tiedostossa DOM-objektin avulla ja tämän elementin sisältö pävitetään string-merkkijonolla.

```
 1: function process() {
 2:   var string = "<ul>"
 3:   + "<li>Green</li>"
 4:   + "<li>Yellow</li>"
 5:   + "<li>Blue</li>"
 6:   + "</ul>";
 7:
 8:   myDiv = document.getElementById("divElement");
 9:   myDiv.innerHTML = string;
10: }
```

Kuva 2 jsFunctions.js, JavaScript-tiedosto

Internet-selaimessa index.html tulostaa seuraavanlaisen sisällön:

List of colors:
– Green
– Yellow
– Blue
(Darie, ym. 2006, 35,36)

2.6 JQuery

JQuery on JavaScript kirjasto, joka tarjoaa helpon ja nopean tavan muokata web-sivua. JQuery:n avulla saadaan helposti aikaan näyttävää ja modernia toiminnallisuutta. JQuery tarjoaa uuden tavan kirjoittaa JavaScriptiä.

JQuery on suhteellisen tuore (2006) tapa muokata dokumenttia, valita DOM-elementtejä, luoda animaatioita, hallita muutoksia sekä kehittää Ajax -sovelluksia.

JQuery on kehittäjien suosiossa ja sille löytyykin lukuisia erilaisia lisäosia, joiden käyttötarkoitukset ja mahdollisuudet ovat lähes rajattomat. Näiden lisäosien avulla jokainen web-ohjelmoinnin perusteet ymmärtävä kykenee luomaan näyttävää visuaalista sisältöä pienellä vaivalla. (JQuery – Wikipedia 2010; jquery.com 2010)

JQueryn käyttöönotto tapahtuu lataamalla erillinen jquery.js-kirjasto, joka linkitetään web-sivulle, kuten mikä tahansa ulkoinen tiedosto; ottamalla se käyttöön <head> -tagin sisällä:

```
<head>
<script type=”text/javascript” scr=”jquery.js”></script>
</head>
```

3 AJAX

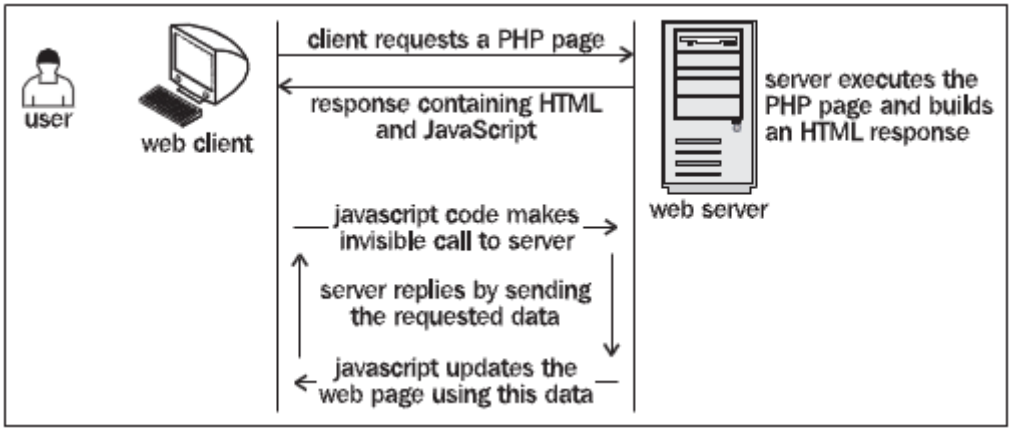
Alunperin nimeämiseen käytettiin termiä XMLHttpRequest, joka kuitenkin oli turhan monimutkainen ja vaikeasti lausuttava käsite esitettäväksi asiakkaille. Niinpä vuonna 2005 oman konsultointiyrityksen omaava Jesse James Garret päätti käyttää termiä Ajax, josta tulikin yleisemmin tunnettu ja käytetty. (Videolähde, 2008, AJAX essential training, what is ajax)

Ajax tulee sanoista Asynchronous JavaScript and XML. Lyhyesti avattuna tällä tarkoitetaan, että Ajax on ”kehittynyt JavaScript”. Se tarjoaa tekniikan, jolla client-puolella JavaScript voi tehdä kutsuja palvelimelle ja päivittää sivuston tietoja ilman erillisiä sivun latauksia. Oikein käytettynä Ajax tarjoaa Internet-sivuston käyttäjälle entistä vuorovaikutteisemman, nopeamman ja käytettävämmän kokonaisuuden. (Darie, ym. 2006, 14,15)

Ajax -tekniikan tarkoitus on saavuttaa parempi kommunikointi client- ja palvelinpuolen toiminnallisuuksille, kun suoritetaan käyttäjän tuottamia pyyntöjä.

Ennen Ajax:n tuomaa tekniikkaa asiakkaan pyyntöihin liittyvä data oli mahdollista saada palvelimelta tai lähettää palvelimelle vain päivittämällä selain ja lähettämällä kaikki pyyntöön liittyvä tieto samalla kertaa. Ajax osaa jakaa pyynnöt osiin ja käsittelee dataa taustalla ilman käyttäjän toimia.

Käyttökohteita Ajax:in käytöstä olisi esimerkiksi lomakkeen tietojen validointi ilman tietojen lähetystä palvelimelle, Google Suggest-tyylinen automaattinen hakusanojen täydennys ja tietokannasta haettava tietotaulukko, johon voi lisätä tai josta voi poistaa tietoja ilman tietojen lähetystä palvelimelle ja sivun uudelleen lataamista. Lomakkeen yksittäisen kentän tietojen tarkastus voidaan toki toteuttaa myös pelkällä JavaScriptillä, mutta Ajax:ia tulee hyödyntää jos halutaan, että esimerkiksi käyttäjän tunnistetiedot tarkistetaan myös palvelimelta ilman, että kaikkia lomakkeen tietoja lähetetään samalla kertaa.



Kuva 3 Tyypillinen AJAX kutsu. Kuva kirjasta AJAX and PHP – Building responsive web applications.

Ajax -tekniikkaa käyttäviä sivuja ei voi helposti laittaa kirjanmerkkeihin, hakukoneet eivät indeksoi kaikkia Ajax -sivuston osia ja JavaScript voi olla pois päältä käyttäjän selaimessa. Nämä seikat puolestaan voidaan lukea Ajax:n haittapuoliksi. (Asleson & Scutta 2007, 37; Darie, ym. 2006, 14-16)

Teknologiat, joista Ajax on tehty, ovat valmiina ja käytössä kaikissa yleisimmissä Internet-selaimissa. Käyttäjän ei siis tarvitse asentaa mitään ylimääräistä, jotta Ajax toimisi. Ajax muodostuu seuraavista tekniikoista:

- JavaScript on kaikkein tärkeimmässä roolissa. Se mahdollistaa client-puolen toiminnallisuuden. JavaScriptin avulla mahdollistetaan DOM-objektin käyttö ja sen kautta HTML-sivun muokkaaminen.
- XMLHttpRequest-objekti mahdollistaa JavaScriptille asynkronisen pääsyn palvelimelle. Asynkroninen tiedonsiirto mahdollistaa tiedon käsittelemisen käyttäjän huomaamatta.
- Palvelinpuolen ohjelmointi on tarpeellista, jotta JavaScriptin tuottamia pyyntöjä voidaan käsitellä ja sisällön saaminen palvelimelta on mahdollista. (Darie ym. 2006, 15)

3.1 Asynkroninen tiedon käsittely

Käyttäjälle huomaamattoman tiedon käsittelyn vaatimuksena on, että tiedon siirto tapahtuu asynkronisesti. Hyvänä esimerkkinä voidaan pitää jo mainittua lomakkeen tietojen täyttämistä: Kun käyttäjä syöttää kenttiin tietoa ja lähettää ne kaikki samalla kertaa palvelimelle, uuden tiedon syöttämisen aloittaa voi heti lähetä-napin painamisen jälkeen. Palvelin käsittelee pyynnön taustalla ja käyttäjä välttyy turhilta odotuksilta ja kokee käyttökokemuksen miellyttävänä.

Tällainen taustalla tapahtuva tietojen käsittely mahdollistaa useita erilaisia käyttötapoja. Esimerkiksi jo mainitun lomakkeen yksittäisten kenttien tietojen tarkistaminen voidaan toteuttaa seuraavalla tavalla: Täytettyjen kenttien tiedot lähetetään palvelimelle tietyin väliajoin validointia varten. Jos tiedoissa on puutteita tai virheitä, voidaan käyttäjää huomauttaa esimerkiksi päivittyvällä tekstillä. Tämä säästää myös palvelinta ylimääräiseltä rasitukselta, koska lomaketta ei tarvitse luoda uudelleen sen virheellisen täyttämisen jälkeen. (Darie ym. 2006, 50)

3.2 Synkroninen tiedon käsittely

Ajax:ia voidaan käyttää myös synkronisesti. Tällöin esimerkiksi edellä mainitun lomaketiedon lähetä-napin painamisen jälkeen ei käyttäjä voisi aloittaa uuden tiedon syöttämistä ennen kuin kaikki vaadittava tieto on siirtynyt asiakkaan ja palvelimen välillä. Synkronista tiedonsiirtoa ei yleisesti hyödynnetä Ajax:in kanssa.

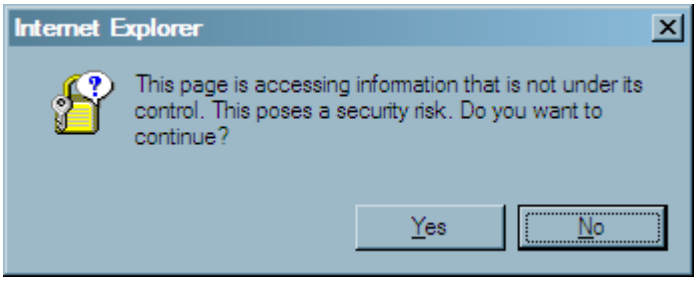
Synkroninen tiedon käsittely olisi kuitenkin hyödyllistä esimerkiksi jos on tarve lähettää peräkkäin monta kyselyä, jotka vaikuttavat toistensa

sisältöön. Tällöin palautetun tiedon saapumisjärjestyksellä on väliä. (Ajax – Asynchronous or Synchronous 2010)

3.3 XMLHttpRequest-objekti

Tekniikka, joka mahdollistaa Ajax:in käytön, on, XMLHttpRequest. Se mahdollistaa JavaScriptin tehdä asynkronisia pyyntöjä palvelimelle. Sen kehitti alun perin Microsoft vuonna 1999 ActiveX-objektina Internet Explorer-selaimeen. Myöhemmin siitä tuli de facto standardi kaikille selaimille ja se onkin natiivina kaikissa nykyisissä selaimissa lukuun ottamatta IE6-selainta. Huolimatta sen yleisestä käytöstä se ei kuitenkaan ole W3C:n standardi. (Darie ym. 2006, 42,43)

XMLHttpRequest-objekti toimii ns. oman selaimen ”hiekkalaatikon” sisällä. Tämä tarkoittaa sitä, että jokaisen pyynnön tulee tapahtua saman verkkotunnuksen eli domainin sisällä, josta komennon suoritus on aloitettu. Tämä turvallisuustoimi on kuitenkin selainkohtainen ja jokaisen selaimen ominaisuuksiin kuuluu tämän kiertäminen, tavalla tai toisella, yleensä vain yhtä nappia painamalla. (Asleson & Scutta 2007, 37,38)



Kuva 4 Internet Explorerin ilmoitus mahdollisesta turvallisuusuhasta

Taulukko 1 XMLHttpRequest-objektin ominaisuudet.

Ominaisuudet	Kuvaus
abort()	Keskeyttää pyynnön
getAllResponseHeaders()	Palauttaa vastauksen otsaketiedot merkkijonona
getResponseHeader(”headerLabel”)	Palauttaa halutun otsakkeen arvon merkkijonona
open(”method”, ”URL”[, asyncFlag[, ”username”[, ”password”]]])	Avaa yhteyden. Metodi voi olla joko GET, POST tai PUT. Pakollisia parametreja ei ole kuin metodi ja url. asynkFlag –boolean arvolla määritellään onko yhteysmuoto asynkroninen (true) vai synkroninen (false)
send(content)	Suorittaa HTTP-pyynnön
setRequestHeader(”label”, ”value”)	Asettaa arvoparin annetulle otsakkeelle
onreadystatechange	Asettaa toiminnon, joka suoritetaan tilamuutoksen tapahtuessa
readyState	Palauttaa tila-arvon: – alustamaton – lataamassa

	– ladattu – interaktiivinen – valmis
responseText	Palvelin palauttaa vastauksen merkkijonona
responseXML	Palvelin palauttaa vastauksen XML-muodossa
status	Palvelimen ilmoittama tilakoodi esim. 200 = OK; 404 = Ei käytettävissä
statusText	Palvelimen ilmoittama tilakoodi merkkijonona

3.4 JSON

Vaikka XML-rakenne on selkeä, on se silti hitaasti ajettava eikä sen päivittäminenkään ole monimutkaisten tietorakenteiden osalta kovin käyttäjäystävällistä.

Vaihtoehdon tarjoaa JSON (JavaScript Object Notation), kieliriippumaton tekstiformaatti, jonka rakenne on samankaltainen C-pohjaisten ohjelmointikielten kanssa. JSON ei varsinaisesti liity juuri Ajax:n käyttöön, mutta web-ohjelmoinnissa sen käyttö ilman joko Ajax:ia tai PHP:ta on harvinaista.

JSON on rakennettu kahden tietorakenteen varaan:
– nimi/arvo-parien joukko (olio, tallenne tai sanalista)
– arvojen järjestetty lista, yleensä toteutettu taulukkorakenteena

Luodaan esimerkkinä customer-olio, joka koostuu etu- ja sukunimestä sekä id-numerosta.

```
var customer =
{
    ”fname” : ”John”
    ,”lname” : ”Doe”
    ,”id_num” : 123
}
```

Olion tietoihin pääsee käsiksi käyttämällä pistenotaatiota:

```
var fName = customer.fname; //luodaan muuttuja fName
var lName = customer.lname; //luodaan muuttuja lName
customer.id_num = 321; //päivitetään asiakkaan id_num
```

Sama customer-olio luotaisiin XML-muotoisena seuraavasti:

```
<customer>
    <firstname>John</firstname>
    <lastname>Doe</lastname>
    <id_num>123</id_num>
</customer>
```

Kuten huomataan, on JSON -esitystapa paljon tiiviimpi, selkeämpi ja muokattavampi. Suurissa tietorakenteissa tämä johtaa huomattaviin suorituskyvyn parannuksiin.
(Asleson & Scutta 2007, 69-71; JSON - Wikipedia 2010.)

4 ESIMERKKI ESILLE TULLEIDEN TEKNIIKOIDEN KÄYTÖSTÄ

Ajax ja palvelin keskustelevat keskenään vain palvelinpuolen ohjelmointikielen välityksellä. Ajax toimii vain clientin Internet-selaimen sisällä ja suorittaa pyyntöjä, joilla palvelimelta haetaan vain pieni määrä tietoa kerrallaan.

Jotta voidaan lähettää tietoa tai saada sitä palvelimelta Ajax:in tulee käyttää XMLHttpRequest-objektia. Saadun tiedon käsittelemiseen on kaksi tapaa responseText ja responseXML. Nimeämisistäkin päätellen responseText palauttaa vastauksen tekstimuotoisena merkkijonona ja responseXML xml-muotoisena. (Asleson & Scutta 2007, 41)

Seuraavassa esimerkissä luodaan Ajax:ia hyödyntävä navigointi. Linkkiä painamalla sitä vastaava sisältö ladataan asynkronisesti sille luotuun div-elementtiin. Linkistä latautuva sisältösivu hyödyntää myös Ajax:ia ja pitää sisällään täytettävän lomakkeen, jonka täytetyt tiedot lähetetään palvelimella sijaitsevaan tietokantaan. Lähetyksen onnistumisesta riippuen, palvelin palauttaa vastauksen, joka tulostetaan sille luotuun div-elementtiin. Esimerkin kaltaista toteutusta tullaan hyödyntämään myös tämän opinnäytetyön pohjalta toteutettavassa sovelluksessa.

Jotta esimerkki pysyisi suhteellisen selkeänä ja lyhyenä siinä ei ole mahdollista käydä läpi kaikkea aiemmin esiteltyä tekniikkaa perin pohjin. Syvempään tutustumiseen Ajax/PHP – aiheisten kirjojen lukeminen on suositeltavaa, ne poikkeuksetta käsittelevät myös JavaScriptiä ja useimmiten jollakin tasolla myös tietokantoja ja HTML:n perusteita.

Esimerkin toteutuksessa hyödynnetyt lähteet. (Asleson & Scutta 2007, 127-139; Darie ym. 2006, 59-61; PHP:PDO - Manual 2010.)

4.1 Etusivu index.html

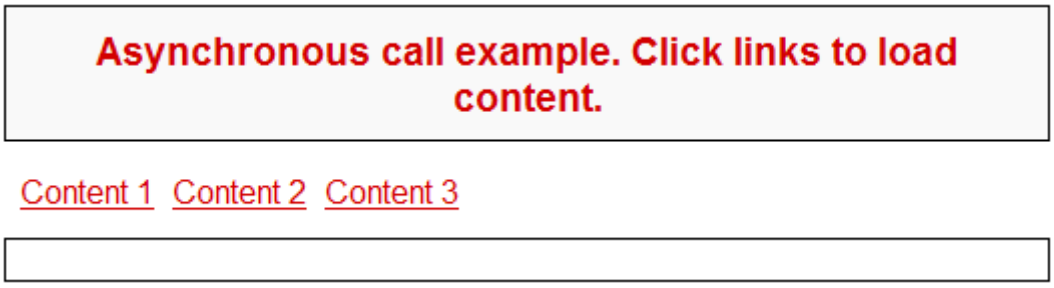
Esimerkissä luodaan html-sivu index.html, jossa on kolme linkkiä, joista kukin kutsuu JavaScript-funktiota getContent(), lähettäen sille url-polun linkkiä koskevaan sisältöön. JavaScript-funktiot ovat erillisessä ajax_contentLoader.js –tiedostossa, joka pitää sisällään myös AJAX-komennot. AJAXin avulla tiedot pyydetään palvelimelta ja ladataan asynkronisesti index.html-sivulla olevaan tyhjään ”ajax_content” div –elementtiin. Sivun ulkoasun muokkausta varten on luotu myös yksinkertainen tyylimäärittelytiedosto styles.css.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=windows-utf8">
    <title>AJAX asynchronous call example</title>
    <script type="text/javascript" language="javascript"
    src="ajax_contentLoader.js"></script>
    <link href="styles.css" rel="stylesheet" type="text/css" />
  </head>

  <body>
    <h1>Asynchronous call example. Click links to load content.
    </h1>
    <table>
      <tr>
      <td>
        <!--send url and target div to .js -->
        <a href="javascript:void(0)"
        onclick="getContent('contents/content1.php');">
        Content 1</a>
      </td>
      <td>
        <a href="javascript:void(0)"
        onclick="getContent('contents/content2.php');">
        Content 2</a>
      </td>
      <td>
        <a href="javascript:void(0)"
        onclick="getContent('contents/content3.php');">
        Content 3</a>
      </td>
      </tr>
    <table>
    <!-- empty div to load server response into -->
    <div id="ajax_content"></div>
  </body>
</html>
```

Kuva 5 Html-tiedosto index.html

```
body{
 font-family:Arial;
 font-size:8pt;
}
h1{
 font-size:14pt;
 color:#D20000;
 font-weight:bold;
 border:solid 1px black;
 padding:10px;
 background-color:#f9f9f9;
 text-align:center;
 width:500px;
}
a{
 font-size:12pt;
 padding-left:5px;
 color:#D20000;
}
#ajax_content{
 border:solid 1px black;
 margin-top: 10px;
 padding: 10px;
 width:500px;
}

```

Kuva 6 Tyylitiedosto styles.css



Kuva 7 index.html Internet-selaimessa

4.2 JavaScript-tiedosto ajax_contentLoader.js

Tiedosto sisältää funktiot:

- sisäsivujen lataukseen (getContent)
- lomakkeen tiedon lähettämiseen/vastauksen saamiseen palvelimelta (doPhpRequest)
- AJAX:in käyttöönottavan (createXmlHttpRequestObject)

Funktio createXmlHttpRequestObject palauttaa XMLHttpRequest-objektin, joka mahdollistaa AJAX:in käytön. Funktio tarkistaa käytettävän selaimen ja sen perusteella palauttaa vaaditun objektin.

```
1  // creates an XMLHttpRequest instance
2  var xmlHttp = null;
3  function createXmlHttpRequestObject()
4  {
5    // this should work for all browsers except IE6 and older
6    try
7    {
8    // try to create XMLHttpRequest object
9    xmlHttp = new XMLHttpRequest();
10   } catch(e)
11   {
12     // assume IE6 or older
13     var XmlHttpVersions = new Array("MSXML2.XMLHTTP.6.0",
14     "MSXML2.XMLHTTP.5.0",
15     "MSXML2.XMLHTTP.4.0",
16     "MSXML2.XMLHTTP.3.0",
17     "MSXML2.XMLHTTP",
18     "Microsoft.XMLHTTP");
19     // try every prog id until one works
20     for (var i=0; i<XmlHttpVersions.length && !xmlHttp; i++)
21     {
22       try
23       {
24         // try to create XMLHttpRequest object
25         xmlHttp = new ActiveXObject(XmlHttpVersions[i]);
26       }catch (e) {}
27     }
28   }
29   // return the created object or display an error message
30   if (!xmlHttp)
31     alert("Error creating the XMLHttpRequest object.");
32   else return xmlHttp;
33 }
```

Kuva 8 JavaScript-funktio createXMLHttpRequestObject

Funktio getContent suoritetaan, kun painetaan jotain navigaation linkkiä. Funktio palauttaa linkin url-polun määrittävän tiedoston sisällön tyhjään div-elementtiin ”ajax_content”. Tiedon siirto tapahtuu GET-metodia käyttäen ja ilman sivun uudelleenlatausta; asynkronisesti.

```
35 function getContent(url)
36 {
37   var xmlHttp = createXmlHttpRequestObject();
38   // only continue if xmlHttp isn't void
39   if(xmlHttp)
40   {
41     // try to connect to the server
42     try
43     {
44       // set true to load asynchronously
45       xmlHttp.open("GET",url,true);
46       xmlHttp.onreadystatechange = function()
47       {
48        if(xmlHttp.readyState == 4) //if done
49         document.getElementById("ajax_content")
50         .innerHTML = xmlHttp.responseText;
51        else //if requested page not found
52         document.getElementById("ajax_content")
53         .innerHTML = 'Broken link';
54       }
55
56       xmlHttp.send(null);
57     } catch (e)
58     {
59       alert("Can't connect to server:\n" + e.toString());
60     }
61   }
62 }
```

Kuva 9 JavaScript-funktio getContent

Funktio doPhpRequest suoritetaan, kun painetaan content2.php –sisältösivun submit-nappia. Funktiossa haetaan sisältösivun tekstikenttien arvot ja lähetetään ne addClient.php-tiedostolle, käyttäen POST-metodia. Funktio lukee palvelimelta saadun vastauksen ja tulostaa sen ”ajax_content”div –elementtiin. Tiedonsiirto tapahtuu asynkronisesti.

```
64 function doPhpRequest()
65 {
66   var xmlHttp = createXmlHttpRequestObject();
67   if(xmlHttp)
68   {
69     try
70     {
71       var name = document.getElementById("name").value;
72       var passwd = document.getElementById("passwd").value;
73       var street = document.getElementById("street").value;
74       var phone = document.getElementById("phone").value;
75       var queryString = "name=" +name+ "&passwd=" +passwd+
76       "&street=" +street+ "&phone=" +phone;
77
78       xmlHttp.open("POST","contents/addClient.php", true);
79       xmlHttp.setRequestHeader("Content-Type",
80       "application/x-www-form-urlencoded");
81       xmlHttp.onreadystatechange = function()
82       {
83        if(xmlHttp.readyState == 4) //if done
84         document.getElementById("ajax_content")
85         .innerHTML = xmlHttp.responseText;
86        else //if requested page not found
87         document.getElementById("ajax_content")
88         .innerHTML = 'Error: client not added to database';
89       }
90       xmlHttp.send(queryString);
91     } catch (e)
92     {
93       alert("Can't connect to server:\n" + e.toString());
94     }
95   }
96 }
```

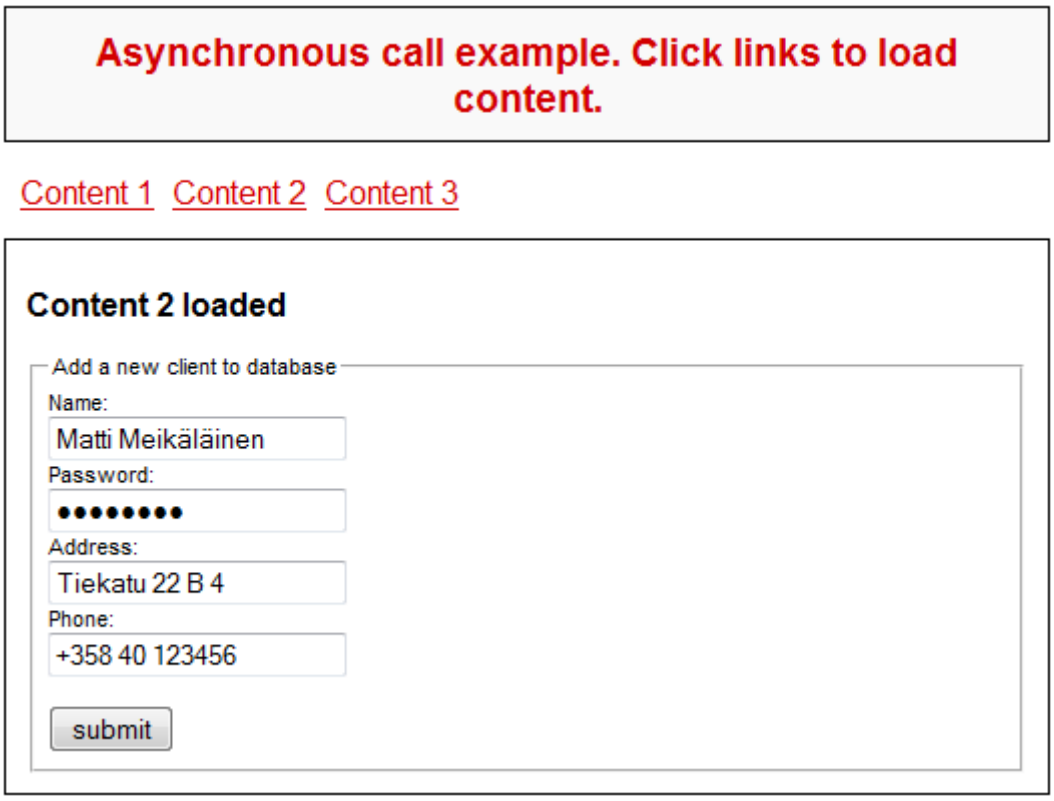
Kuva 10 JavaScript-funktio doPhpRequest

4.3 Sisältösivu content2.php

Php-tiedosto content2.php sisältää täytettävät tekstikentät ja submit-napin, jota klikkaamalla kenttien tiedot lähetetään ajax_contentLoader.js JavaScript-tiedoston avulla palvelimelle.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=windows-utf8">
    <title></title>
  </head>

 <body>
  <h2>Content 2 loaded</h2>
  <div >
    <fieldset id="fieldset">
      <legend>Add a new client to database</legend>

      <label for="nimi">Name:</label> <br class="nobr" />
      <input type="text" id="name" /> <br />

      <label for="nimi">Password:</label> <br class="nobr" />
      <input type="password" id="passwd"/> <br />

      <label for="nimi">Address:</label> <br class="nobr" />
      <input type="text" id="street" /> <br />

      <label for="nimi">Phone:</label> <br class="nobr" />
      <input type="text" id="phone" /> <br />

      <label for="submit"> </label><br class="nobr" />
      <input type='button' onclick='doPhpRequest()'
      value='submit' /> <br />
    </fieldset>
  </div>

 </body>
</html>
```

Kuva 11 Php-tiedosto content2.php



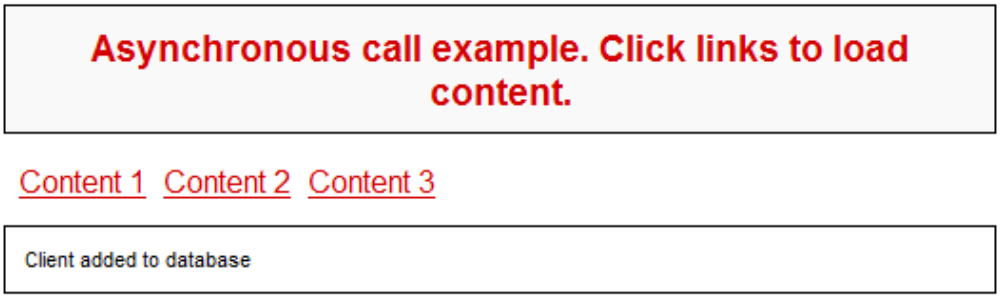
Kuva 12 content2.php Internet-selaimessa

4.4 Php-tiedosto addClient.php

Tiedoston addClient.php avulla täytettyjen tekstikenttien arvot saadaan lähetettyä palvelimella sijaitsevaan tietokantaan. Tietojen lähetykseen käytetään tietoturvaa parantavaa PDO-luokkaa. Lähetyksen onnistuessa palautetaan teksti ”Client added to database”, joka tulostetaan AJAX:in avulla ”ajax_content” div-elementtiin.

```php
<?php
  $hostname = "localhost";
  $dbname = "ajaxexample";
  $username = "root";
  $password = "";

  try{
    $dbh = new PDO("mysql:host=$hostname;dbname=$dbname",
    $username, $password,
    array(PDO::ATTR_PERSISTENT => true));
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch(PDOException $e){
    echo $e->getMessage();
    die();
  }

$dbh->exec('SET CHARACTER SET utf8');

  try {
    $dbh->beginTransaction();
    $query = 'INSERT INTO client SET
      name = :name ,
      passwd = :passwd ,
      street = :street,
      phone = :phone';

    $stmt = $dbh->prepare($query);
    $stmt->bindParam(':name',   $_POST['name'],  PDO::PARAM_STR);
    $stmt->bindParam(':passwd', $_POST['passwd'],PDO::PARAM_STR);
    $stmt->bindParam(':street', $_POST['street'],PDO::PARAM_STR);
    $stmt->bindParam(':phone',  $_POST['phone'], PDO::PARAM_STR);
    $stmt->execute();
    $dbh->commit();

   echo "Client added to database";
    } catch(PDOException $e)
    {
        $dbh->rollBack();
        echo $e->getMessage();
    }
$dbh = null;
?>
```

Kuva 13 Php-tiedosto addClient.php



Kuva 14 Palvelimen palauttama vastaus Internet-selaimessa

5 CASE: FARMEDICO

Esiteltyjä tekniikoita tullaan laajasti hyödyntämään verkkosivustossa, joka toteutetaan lääkealalla toimivalle Farmedico- henkilöstöpalveluyritykselle. Farmedicon palveluihin kuuluvat muun muassa henkilöstön vuokraus ja välitys apteekkeihin sekä lääkealan yrityksiin. Sivustoon toteutetaankin alkuun vain henkilöstövuokrauspalvelu ja mahdollisesti myöhemmässä vaiheessa muuta intranetiin liittyvää toiminnallisuutta.

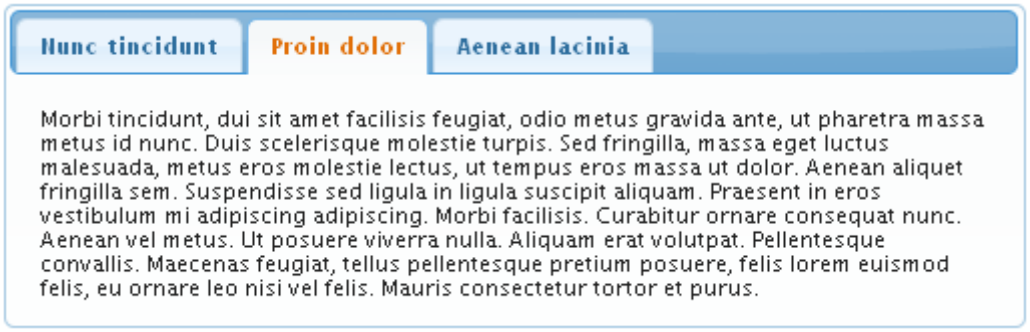
Henkilöstövuokrauspalvelun toteutuksen tarkoitus on saada aiemmin puhelimitse tehty työntekijöiden välitys sähköiseen muotoon Internetiin. Toteutuksen tulee olla selkeä, muokattava sekä helposti käytettävä.

5.1 Perusteet tekniikoiden käytölle

Teoriaosuudessa esille tulleiden tekniikoiden järkevästi suunniteltu yhteiskäyttö mahdollistaa täysin dynaamisen, näyttävän ja helppokäyttöisen web-sovelluksen toteuttamisen. Näitä ominaisuuksia useat Internetiä käyttävät arvostavat ja myös Farmedico toivoo, että nämä ominaisuudet yhdistyvät sovelluksessa. Sovelluksen toteuttamisen kannalta pakollisia tekniikoita ovat vain html, php ja tietokannat/xml-tiedostot, mutta muiden tekniikoiden mukaan tuominen antaa täysin erilaiset näkökulmat, mahdollistaen dynaamisen ja näyttävän toteutuksen Sovellus tullaan toteuttamaan pääasiassa asiakkaiden näkökulmasta, samalla kuitenkin huomioiden myös muut käyttäjät.

Kuten todettua, JQueryn ja AJAX:in käyttö ei ole pakollista toteutuksen kannalta. AJAX ja sen tuomat hyödyt tulivat hyvin esille edeltävässä esimerkissä, jossa luotiin AJAX:ia hyödyntävä navigointi. JQuery:n käyttö on perusteltua sen mahdollistavien ominaisuuksien mukaan. Näitä ominaisuuksia ovat muun muassa Datepicker-komponentti, jonka avulla halutun päivän valitseminen on helppoa ja selkeää sekä Tabs-komponentti, jonka avulla aiemmin AJAX:in avulla toteutettu navigaatio mahdollistetaan vain muutamalla rivillä koodia. Nämä komponentit kuten sadat muutkin ovat ladattavissa JQueryn kotisivuilta. JQuery siis vähentää työn määrää huomattavasti selkeyttäen samalla koodia, mahdollistaen helpomman muokkauksen tulevaisuudessa.

Kuva 15 JQuery Datepicker-komponentti



Kuva 16 JQuery Tabs-komponentti

5.2 Toiminnallinen määrittely

Sivusto tulee jakautumaan kolmeen eri osioon. Osiot ovat hallintapuoli, asiakaspuoli sekä työntekijäpuoli. Käyttäjällä on ryhmästä riippuen pääsy vain yhteen osioon, tarkoittaen, että samalla tunnuksella ei voi kirjautua kuin sille ennalta määrättyyn osioon. Käyttäjä kirjautuu sisään omilla tunnuksillaan ja ohjautuu ennalta asetetun ryhmä-id-arvon mukaan sille määrättyyn osioon. Kirjautumiseen käytetään erillistä sivua, jossa on tekstikentät käyttäjätunnukselle ja salasanalle. Kirjautumistiedot tulee tarkistaa MySQL-tietokannasta. Salasanan tulee olla salattu jollain salausalgoritmilla, kuten MD5:lla.

Hallintapuolen toiminnallisuuksiin kuuluvat käyttäjien ja asiakkaiden lisäys, muokkaus ja poistaminen, syötettyjen työvuorojen muokkaaminen, poistaminen sekä erimuotoinen listaaminen, kuten vapaiden työvuorojen listaus. Hallintapuolella tulee olla mahdollista seurata kaikkea vuokraukseen liittyvää toimintaa ja näin tarkkailla tapahtumia sekä puuttua mahdollisiin väärinkäytöksiin tai vahingossa väärin syötettyyn ilmoitukseen.

Asiakaspuolen toiminnallisuuksiin kuuluvat työvuorojen lisäys, sekä vuoroihin ilmoittautuneiden työntekijöiden tietojen näkeminen, mahdollista yhteydenottoa varten. Vielä tässä vaiheessa epäselvää on, toteutetaanko järjestelmä, joka lähettäisi asiakkaalle aina sähköpostin, kun

työntekijä varaa vapaan vuoron. Sähköposti sisältäisi työvuoron sekä työntekijän tiedot.

Työntekijäpuolella voi listata kaikki asiakkaiden luomat vapaat vuorot ja ilmoittautua haluamalleen vuorolle sekä listata vuorot joihin on jo ilmoittautunut työntekijäksi. Työntekijöillä ei ole mahdollista perua jo varaamaansa vuoroa, eikä työntekijä myöskään näe muiden työntekijöiden varaamia vuoroja. Työntekijän on mahdollisuus kirjata jokaisen päivän kohdalle erikseen mikäli hän on vapaa töihin. Tämä tapahtuu osiossa, jossa hän näkee vuorot, joihin on ilmoittautunut. Nämä tiedon myös työnantaja eli Farmedico näkee hallintapuolelta.

Ti 04.04	**En pääse töihin**	(muokkaa)	
Ke 05.04	**En pääse töihin**	(muokkaa)	
To 06.04	**En pääse töihin**	(muokkaa)	
Pe 07.04	**En pääse töihin**	(muokkaa)	
La 08.04	**10:00-18:00**		Hämeenkadun apteekki
Su 09.04		(muokkaa)	
	VIIKKO 13		
Ma 10.04	**09:00-17:00**		Hämeenkadun apteekki
Ti 11.04	**09:00-17:00**		Hämeenkadun apteekki

Taulukko 2 Työntekijälle kalenterimainen listaus omista vuoroistaan

Palvelu tulee toteuttaa niin, etteivät päällekkäisvaraukset ole mahdollisia. Työntekijän varattua työvuoron sen tulee poistua vapaiden vuorojen listalta. Suurten tietomäärien vuoksi toteutuksessa tullaan hyödyntämään MySQL-tietokantaa.

5.2.1 Työvuorojen lisäys sekä valitseminen

Asiakkaiden lisätessä uuden vuoron, tulee päivämäärän olla helposti valittavissa. Valinta tapahtuu erillisestä kalenterista, kuten tämän tyylisissä ajanvaraussovelluksissa on tapana. Asiakkaiden tulee myös nähdä itse omat laittamansa vuorot listana, joka on jaoteltu viikottain, mikä tarkoittaa, että jokainen viikko on eritelty näkyvästi toisistaan.

Työntekijöiden työvuorojen listauksen näkymä on samanlainen verrattaessa asiakkaiden näkymään, jossa vuorot näkyvät viikottain. Poikkeuksena tietenkin on, että työntekijät näkevät kaikkien asiakkaiden vapaat vuorot ja asiakas näkee vain omansa. Työntekijä varaa vuoron klikkaamalla tätä hiirellä. Vuoron varaus tulee vielä varmistaa erillisellä ikkunalla, jolloin vältytään mahdollisilta vahinkovarauksilta.

VIIKKO 13	
Ma 10.04	
09:00-16:30	Hämeenkadun apteekki
08:00-16:30	Mannerheimintien apteekki
09:30-17:30	Hämeenkyrön apteekki
12:00-16:30	Männistön apteekki
09:00-16:30	Apteekki Matti Oy
Ti 11.04	
09:00-16:30	Hämeenkadun apteekki
08:00-16:30	Mannerheimintien apteekki
09:30-17:30	Hämeenkyrön apteekki
12:00-16:30	Männistön apteekki
09:00-16:30	Apteekki Matti Oy

Taulukko 3 Työntekijälle näkymä vapaista vuoroista

5.3 Sivutilantarjoajan tekniset ominaisuudet

Sivusto tullaan sijoittamaan samalle palvelimelle, jossa Farmedicon nykyiset sivut ovat. Sivutilantarjoajana toimii nettihotelli.fi, jonka PHP-versiona toimii yleisin uusi 5.2 -sarjalainen. Tietokantana käytössä on MySQL ja sen hallintatyökaluna toimii phpMyAdmin. Farmedicon nykyinen sopimus nettihotelli.fi:n kanssa vaatii, että tietokanta tulee tilata erikseen.

5.4 Sivuston käytettävyys

Sivuston käytettävyyteen kiinnitetään erityistä huomiota. Käyttäjän tarvitsemat toiminnot ovat helposti esillä ja sivusto on niin kevyt, ettei sen lataaminen häiritse käyttäjää. Sivusto toteutetaan käyttäen hyväksi AJAX:in tuomaa hyötyä, jonka avulla saadaan kaikki lataaminen tapahtumaan asynkronisesti ja käyttäjän huomaamatta. Kun käyttäjä valitsee uuden välilehden, ei tapahdu näkyvää sivun latausta, vaan uusi sisältö tulee sille varattuun tilaan, samoin tavoin kuin esimerkiksi työpöytäohjelmissa. Tämänkaltainen toteutus on nähtävissä kappaleessa, jossa käsiteltiin esimerkillä teoriaosuudessa esille tulleita tekniikoita. Ulkoasun suunnittelussa huomioidaan selkeys ja käytettävyys. Ulkoasu tulee noudattamaan Farmedicon yleistä linjaa.

Käytettävyyden kannalta huomioitavaa on myös, että sivusto toimii kaikilla yleisimmillä selaimilla. Näitä ovat Internet Explorer, Mozilla Firefox, Opera sekä Safari. Tulee kuitenkin ymmärtää, että vanhat selaimet, kuten Internet Explorer – versio 6 ei tue kaikkia ominaisuuksia ja tätä varten ei sivustoja enää kannata rakentaa.

5.5 Sivuston päivitettävyys

Jo sivuston suunnitteluvaiheessa tullaan huomioimaan sen myöhemmässä vaiheessa tapahtuva päivittäminen. Koodit jaetaan toimintojen mukaan järkevästi osiin, jolloin hallittavuus helpottuu. Usein käytetyt funktiot

toteutetaan erillisiin luokkiin, ja niitä kutsutaan sieltä, tällöin säästytään turhalta toistolta ja mahdollisen muokkauksen joutuu tekemään vain yhteen paikkaan. Huomioon otetaan myös palvelintilan mahdollisesti tapahtuva PHP-version päivittyminen. Päivitettävyyttä helpottaa myös selkeästi toteutettu koodi sekä sen kommentointi. Varsinaista dokumentaatiota ei tulla tekemään, joten kommentointi on hyvin tärkeää.

5.6 Sivuston testaus

Sivusto tullaan ennen julkaisua testaamaan rajatulla käyttäjäryhmällä. Testauksessa tulee huomioida:

- mahdolliset päällekkäisvaraukset
- toimivuus kaikilla yleisimmillä selaimilla
- mahdolliset tietoturva-aukot, turvallisuus
- tietojen oikeellisuus
- kaikkien osa-alueiden toiminta

Kannattavinta olisi, että sivuston testaukseen käytettäisiin ryhmää, joka sitä tulee jatkossakin käyttämään. Tässä tulee kuitenkin muistaa painottaa, ettei kyseessä ole täysin valmis ns. release-versio, jotta säästytään ylimääräisiltä kommenteilta. Varsinaiset käyttäjät antaisivat kuitenkin paljon tarkempaa tietoa kehityksen kannalta kuin muut.

5.7 Sivuston jatkokehitys

Yritysten ilmeeseen sopivia järjestelmiä on hankala löytää. Sovellus, jonka ilmeen kustomointi pienellä vaivalla olisi yrityksille taloudellisesti järkevämpi investointi kuin vastaavat tekeminen alusta asti. Sovelluksen suunnittelu, niin, että sen kustomointi olisi helppoa, on haasteellista ja vaatii huolellisuutta. Jotta sovellus olisi todella kiinnostava, tulee sen myös sisältää muita osioita. Näihin voisi lukeutua esimerkiksi Intranet ja yrityksen sisäinen kuvapankki/keskustelu/blogi-osio.

Farmedicolla onkin toiveena, että myöhemmässä vaiheessa toteutetaan Intranet-palvelu, jonka avulla voidaan jakaa vain tietylle käyttäjäryhmälle tarkoitettuja materiaaleja. Toiveena on myös, että käyttäjien osiota laajennettaisiin niin, että siellä olisi nähtävissä muun muassa palkkakuitit. Toiveena on siis jonkinasteinen tietopankkiratkaisu.

6 POHDINTA

Dynaaminen sisältö Internetissä on lähivuosina kasvanut räjähdysmäisesti. Taitavia ohjelmistosovelluskehittäjiä kouluttautuu koko ajan lisää ja heidän taitojaan hyödynnetään yhä usemmin muuallakin kuin vain työpöytäsovellusteknisissä ratkaisuissa. Internet on kuitenkin sektori, jolla asiakkaat haluavat esitellä tuotteitaan ja mahdollisesti myös myydä niitä kuluttajille. Nämä Internet-sovellukset vaativat sellaisen tekniikan hallitsemista, jota peruskoodarilla ei ole.

Sovellusten kehittäminen on myös tietoturvasyistä haastavaa, sillä huolimattomasti toteutettu tietoturva ja sovellus ovat yhtälö, jonka ei tulisi olla todellinen. Web-palveluja tarjoavat yritykset yleensä toimivat kuitenkin tiukkojen aikataulujen rajoissa ja kehittäjät joutuvat tekemään ratkaisuja, jotka eivät välttämättä ole oikeaoppisia. Tällaiset pienet tekijät saattavat olla ratkaisevia, silloin kun niillä oikeasti on merkitystä. Suurten yritysasiakkaiden budjetit ovat yleensä suurempia ja sovellusten kehittämiseen annetaankin enemmän aikaa. Tällöin myös työn jälki on yleensä parempaa. Tärkeintä kuitenkin on tuntea tekniikat, joilla sovellusta on toteuttamasta. Oli kyseessä Java, PHP, Flash tai muu sovellus, niin aina ratkaisevin merkitys on sillä, kuinka hyvin kehittäjä tietää mitä on tekemässä ja mihin käytettävä tekniikka kykenee. Esimerkiksi Flashilla itsellään on täysin turha yrittää tietoturvallisen kirjautumisen toteuttamista, mutta yhdessä esimerkiksi tietokannassa sijaitsevien käyttäjätunnusten, salasanojen ja PHP:n kanssa tällainen olisi mahdollista.

Perusteet dynaamisten sovellusten toteuttamiseen verkkoon selviävät tämän opinnäytetyö lukijalle. Suhteellisen tuoreet tekniikat JQuery ja Ajax tulevat olemaan esillä suurenevissa määrin tulevien vuosien Internet palveluissa eikä näiden aiheiden sivuuttaminen tullutkaan edes kysymykseen. Ajax:in käyttöönottoa helpottaa myös se, että vaikka se käsitteenä on uusi, tekniikkana ei. Kuten kerrottua se on vain monen vanhan tekniikan yhdistelmä. JQuery puolestaan on vain kehitetty JavaScript ja PHP yhdessä HTML:n CSS-tyylitiedostojen kanssa on kaikille tuttua. Miksi siis ei käyttäisi näitä tekniikoita luotaessa dynaamista Internet-sovellusta. Kaikille tutut tekniikat ja vähän uutta opittavaa. Mitä voi mennä vikaan? Vastaus tähän kysymykseen on yksinkertainen ja yhdellä sanalla ymmärrettävissä; ”kaikki”.

Vaikka käytettävät tekniikat ovat tuttuja ja perusteet ovat hallussa, se ei riitä. Jos sinulta kysytään tekisitkö dynaamisen verkkokauppasovelluksen ja tämän opinnäytetyön luettuasi raavit päätä ja mietit lukemaasi, vastaa ettet tällä kartaa tartu tilaisuuteen. Vaikka tämä opinnäytetyö kertookin, mitä tekniikoita voit käyttää se ei kuitenkaan kerro miten ja milloin niitä on järkevintä käyttää.

LÄHTEET

Asleson, R. Schutta, N. 2006. Ajax Tehokas hallinta, Gummerus kirjapaino oy.

Darie, C. Brinzarea, B. Chereches-Tosa, F. & Bucica, M. 2006. AJAX and PHP: Building responsive web applications, Olton Birmingham: Packt Publishing Ltd.

Souders S. 2007. High performance web sites, O'Reilly Media, Inc.

Internet – Ajax Asynchronous or Synchronous, 2010. Viitattu 24.04.2010. http://javascript.about.com/od/ajax/a/ajaxasyn.htm

Internet – JQuery, 2010. Viitattu 21.03.2010 päivitetty 18.03.2010. http://en.wikipedia.org/wiki/JQuery

Internet – JQuery, 2010. Viitattu 21.03.2010 http://jquery.com/

Internet – PHP:PDO – Manual, 2010. Viitattu 19.04.2010. http://php.net/manual/en/book.pdo.php

Internet – PHP Usage Statistics, 2010. Viitattu 08.04.2010. http://trends.builtwith.com/framework/PHP

Internet – PHP-tietoturvaopas Sigmatic Oy 2010. Viitattu 24.04.2010. http://www.sigmatic.fi/asiakassivut/php-tietoturvaopas.html

Internet – TIOBE Software : Tiobe in-dexhttp://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Internet – Wikipedia, 2010. Viitattu 21.03.2010. http://en.wikipedia.org/wiki/JSON

Internet – Wikipedia, 2010. Viitattu 23.03.2010. http://fi.wikipedia.org/wiki/Internet

Internet – W3C Document Object Model, 2010. Viitattu 24.04.2010. http://www.w3.org/DOM/

Internet – W3C CSS, 2010. Viitattu 14.04.2010. http://www.w3.org/Style/CSS/

Videolähde – Lynda.com, 2008, AJAX Essential Training. http://www.lynda.com/home/DisplayCourse.aspx?lpk2=480

Liite 1

SÄHKÖINEN LIITE

Kappaleen neljä esimerkin tiedostot.